MORAVSKÁ VYSOKÁ ŠKOLA OLOMOUC, o.p.s.

BAKALÁŘSKÁ PRÁCE

2012 Petr Svozilík

MORAVSKÁ VYSOKÁ ŠKOLA OLOMOUC, o.p.s.

Ústav informatiky

Petr Svozilík

Softwarová ekonomicko-manažerská výbava mobilních operačních systémů

Economics and Managerial Software Equipment of Mobile Operating Systems

Bakalářská Práce

Vedoucí práce: PhDr. Jan Lavrinčík, DiS.

Olomouc 2012

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci na téma „Softwarová ekonomicko-manažerská výbava mobilních operačních systémů“ vypracoval samostatně pod vedením pana PhDr. Jan Lavrinčíka, DiS.

V Olomouci…… Petr Svozilík..……..

Poděkování

Úvodem bych chtěl poděkovat vedoucímu této práce PhDr. Jan Lavrinčíkovi, DiS. za cenné rady a vedení bakalářské práce.

Obsah

Praktická část

Úvod

Hlavním důvodem, proč jsem si vybral téma ”Softwarová ekonomicko-manažerská výbava mobilních operačních systémů” je, že mě informační technologie a tato problematika vždy zajímala. Operační systém Google Android je v současné době velmi populárním a nejrozšířenějším mobilním systémem, který potvrzuje v průzkumu analytická společnost Gartner (1). Na mobilním trhu se setkáme i s jinými softwarovými platformami. Například mezi dalšími velmi populárními a rozšířenými platformami jsou iOS od společnosti Apple nebo Symbian od společnosti Nokia. Oba systémy vypadají funkčností velice zajímavě, ale nevlastním mobilní telefon s daným operačním systémem a z tohoto důvodu je nebudu ve své práci dále rozpracovávat. Google Android umožňuje uživateli oproti jiným systémům značnou svobodu při vývoji a rozvoji aplikací používaných v těchto rovinách: open-source platformy, ukázky zdrojových kódů a vývojového prostředí. Software, který nabízí Google Android je šířen prostřednictvím internetu, odkud si můžeme stáhnout potřebné nástroje pro vývoj aplikací. Vývojové prostředí a pluginy od Googlu jsou zcela volně šiřitelné. V Androidu najdeme základní předinstalované aplikace kde jednou z nich je Android Market. Tato služba od Googlu umožnuje uživatelům stahovat různé aplikace do telefonu.

Současný stav problematiky těchto aplikací spočívá v nepřehlednosti a složitosti softwaru. Velká část aplikací postrádá podporu starších verzí operačního systému Google Android, což znemožňuje jejich používání na starších telefonech. Další fakt je, že některé aplikace jsou zbytečně náročné na systém a to může mít například za následek snížení kapacity baterie u telefonu. Proto jsem se rozhodl učinit návrh nové aplikace, která bude jednoduchá, přehledná, nebude zatěžovat systém telefonu a bude se všemi verzemi Google Android kompatibilní.

Práci jsem rozdělil na teoretickou a praktickou část z důvodů přehlednosti navrhovaného řešení. Cílem teoretické části je obecné seznámení s platformou Google Android. V této části podrobně popíši z čeho se skládá Google Android, její architekturu a jak funguje. Dále bych chtěl práci obohatit o historii verzí Google Android. Následně se budu zabývat základními prvky pro tvorbu aplikací, kde vysvětlím Android (SDK) a jeho vývojové prostředí. Systém Google Android se skládá ze základních stavebních komponentů, které v práci podrobně rozeberu - jsou to aktivity, dodavatelé obsahu, služby a záměry. Ve třetí kapitole budu porovnávat nejstahovanější manažerské aplikace pro platformu Google Android, následně je podrobím analytickým testům a vyhodnotím. Dále učiním návrh aplikace, kde si stanovím, jak bude poznámkový blok vypadat a jaké bude mít funkce.

Cíl praktické části je zaměřen na praktiky programování pro platformu Google Android, zejména na principy a postupy uplatněné při programování aplikace „Quick Book“. Pro názornost a přehlednost je práce doplněna ukázkami zdrojových kódů. Dále se zabývám založením Android projektu a přehledu souborů v Eclipsu a popisuji samotný vývoj aplikace aktivit a tříd: NoteBookActivity, EditNoteActivity, ViewNoteAcitivy, NoteBookDbAdapter, Listenery, kde vysvětlím k čemu slouží a jakou hrají roly uvnitř aplikace.

Jako metodiku jsem použil odbornou literaturu týkající se tématu, čerpání z internetových zdrojů zde zmíněných. K analýze aplikací byl použit internetový obchod Android Market.

1. Historie Google Android

Android je open source platforma pro mobilní zařízení, která je založená na pozměněném linuxovém jádře. Systém byl původně vyvíjen firmou Android Inc., kterou v roce 2005 koupil Google. V roce 2007 pak byla založena Open Handset Alliance skládající se z několika desítek firem – kromě Googlu to jsou převážně výrobci mobilních telefonů, polovodičových součástek a také mobilní operátoři. OHA zastřešuje vývoj otevřených standardů pro mobilní zařízení a má za úkol kooperovat vývoj operačního systému Android (2).

1.1 Počátky Androidu

Historie operačního systému Android začala již v roce 2003, kdy stejnojmennou společnost založili Andy Rubin, RichMiner, Nick Sears a ChrisWhite. Jejich cílem bylo, dle jejich slov začít vytvářet chytřejší mobilní přístroje, které budou brát v úvahu nároky uživatelů a jejich polohu. Činnost této společnosti zpočátku nevzbuzovala příliš pozornosti, pro svět byli „jen“ jedním z dalších vývojářů softwaru pro mobilní telefony. Zlom nastal až v roce 2005, kdy společnost Android Inc. byla koupena gigantem Google. Klíčoví lidé na svých postech zůstali i po akvizici (Andy Rubin je nyní vice- prezidentem mobilní divize Googlu) a pod novým majitelem dostal vývoj rychlejší tempo. Objevily se první spekulace o tom, že Google plánuje představit svůj vlastní mobilní telefon (přezdívaný gPhone), což bylo umocněno tím, že získali mnoho patentů v oblasti mobilní komunikace. V roce 2007, jen pár měsíců poté, kdy byl na trh uveden v mnoha směrech revoluční iPhone od Apple, ale Google představil mnohem ambicióznější plán. Společně s dalšími hráči v oblasti mobilní komunikace vytvořili konsorcium Open Handset Alliance, které představilo světu nový mobilní operační systém Android. Mezi zakládajícími členy byli kromě Google například Nvidia, Samsung, LG, HTC, Motorola, Intel, Qualcomm, Ebay, T-Mobile, Telefonica a mnozí další. Jejich společným cílem je podpora a vývoj nově uvedeného mobilního operačního systému, který je založen na otevřených standardech. Aktuální seznam členů naleznete přímo na oficiálních stránkách. I když je od té doby Android zastřešován právě Open Handset Alliancí, hlavní úlohu při vývoji má stále Google. Prvním mobilním telefonem s Androidem původně ve verzi 1.0, který se reálně dostal do prodeje (bylo to na konci roku 2008), se stal T-Mobile G1, který vyráběla společnost HTC (známý také pod kódovým označením HTC Dream). Tento dotykový telefon s hardwarovou qwerty klávesnicí tak odstartoval éru obrovských úspěchů na trhu smartphonů (2).

1.2 Open source platforma (licence)

Open-source označuje koncept jako způsob vývoje a distribuce, který každému potencionálnímu zájemci umožňuje přístup ke zdrojovým datům produktu, kterým může být například program, či hardware. Největší výhodou open-source systému je možnost volně upravovat a vyvíjet daný produkt v rámci komunity mnoha uživatelů - vývoj je tak obvykle mnohem rychlejší a v podstatě nepřetržitý v porovnání s komerčně vyvíjenými produkty - zde je totiž snaha každé zlepšení zpeněžit a tudíž je firmy uvolňují postupně. Navíc nikdo nerozumí svým potřebám a nárokům jako sám uživatel. Programy s licencí open-source podle Open Source Initiative (OSI) musí být, mimo jiné, volně distribuované a jejich zdrojový kód (ve své původní formě) snadno přístupný. Důvodem je snaha zjednodušit proces vývoje daného kódu. Samozřejmě nesmí docházet k žádné diskriminaci určité skupiny uživatelů nebo k vymezení či omezení oblasti využití dané open-source aplikace. Licenční podmínky mají být nezávislé na konkrétních technologiích či rozhraních, nesmí zasahovat do právních vztahů k jiným programům (se kterými jsou například distribuovány), ani být určeny pouze pro konkrétní softwarový produkt (3).

2. Platforma Google Android

Android je založený na Linuxovém jádře pro mobilní telefony s operační systémem vyvinutý od společností Google. Výrobci telefonů nemusí za používání Androidu platit, což znamená snížení ceny telefonu (4). Android také nabízí v sobě předinstalované aplikace šité na míru, které jsou nutností pro běh telefonu. Ve spojení s mobilními aplikacemi, které si můžeme stáhnout do zařízení, představuje tato platforma obrovskou flexibilitu použití.

2.1 Výzvy vývoje aplikací pro chytré telefony

Mezi nesporné výhody patří, že jsou chytré telefony velmi přitažlivé pro uživatele. Počátek internetových služeb v mobilních zařízeních se datuje již od poloviny devadesátých let a od vytvoření jazyka HandheldDeviceMarkupLanguage (HDML). Avšak skutečně se telefony schopné připojení k internetu prosadily až v posledních několika letech. Díky trendům jako je psaní textových zpráv, nebo zařízení iPhone společností Apple, získávají telefony prostřednictvím nichž se lze připojit k internetu, rychle na popularitě. Vývoj aplikací pro Android tedy představuje zkušenost s velmi

zajímavou technologií (Android) na velmi rychle se rozvíjejícím trhu (telefonů umožňujících připojení k internetu) a to za to stojí. Problém nastává ve chvíli, kdy dojde na samotné programování. Každý vývojář, který má nějakou zkušenost s programováním aplikací pro PDA zařízení či telefony, pocítil na vlastní kůži nevýhody plynoucí jednoduše z toho, že jsou telefony ve všech možných ohledech prostě *malé*; mezi tyto nevýhody pak patří například následující omezení:

- Telefony mají velmi malou obrazovku.
- Klávesnice, pokud jimi zařízení disponují, jsou také velmi malé.
- Polohovací zařízení, pokud existují, jsou strašná (jak vám potvrdí každý, kdo ztratil elektronické pero) nebo nepřesná (velké prsty na displeje schopné reagovat na více dotyků najednou nejsou dobrá kombinace).
- Rychlost CPU a velikost paměti jsou v porovnání se stolními počítači a servery velmi omezené.

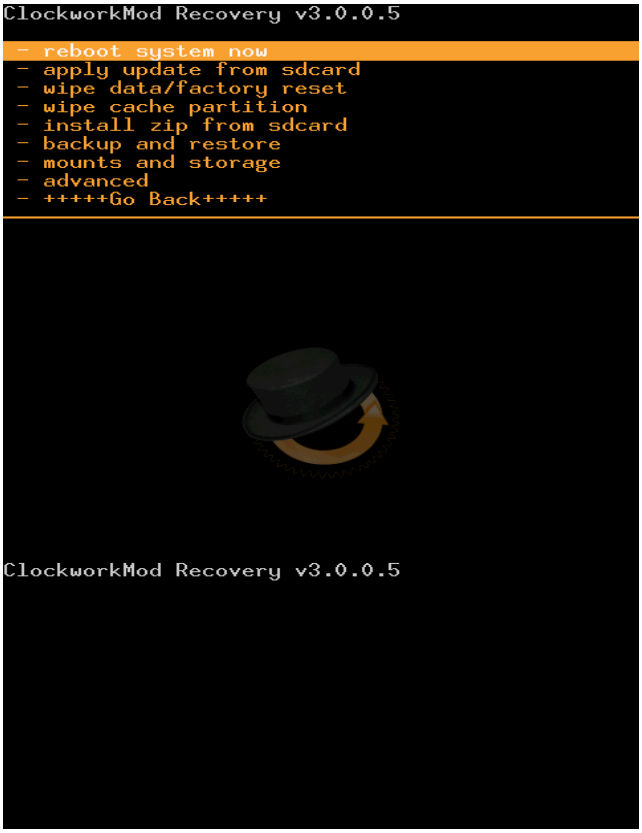
Lidé s mobilními telefony bývají podrážděni, když jejich telefony nefungují. A podobně se uživatelům příliš nelíbí, když jejich telefon přestane fungovat kvůli vaší aplikaci, například z následujících důvodů: Vaše aplikace vytíží procesor telefonu takovým způsobem, že telefon není schopný přijmout příchozí hovor. Vaše aplikace způsobí zhroucení operačního systému, například v důsledku masivních úniků paměti. Z těchto důvodů se tedy programů pro telefony liší od vývoje aplikací pro stolní počítače, webových stránek nebo obslužných serverových procesů. Liší se vývojové nástroje, jinak se chová aplikační rámec a funkce vašich programů podléhají mnoha různým omezením (5).

2.2 Architektura

Bootloader (Zavaděč):

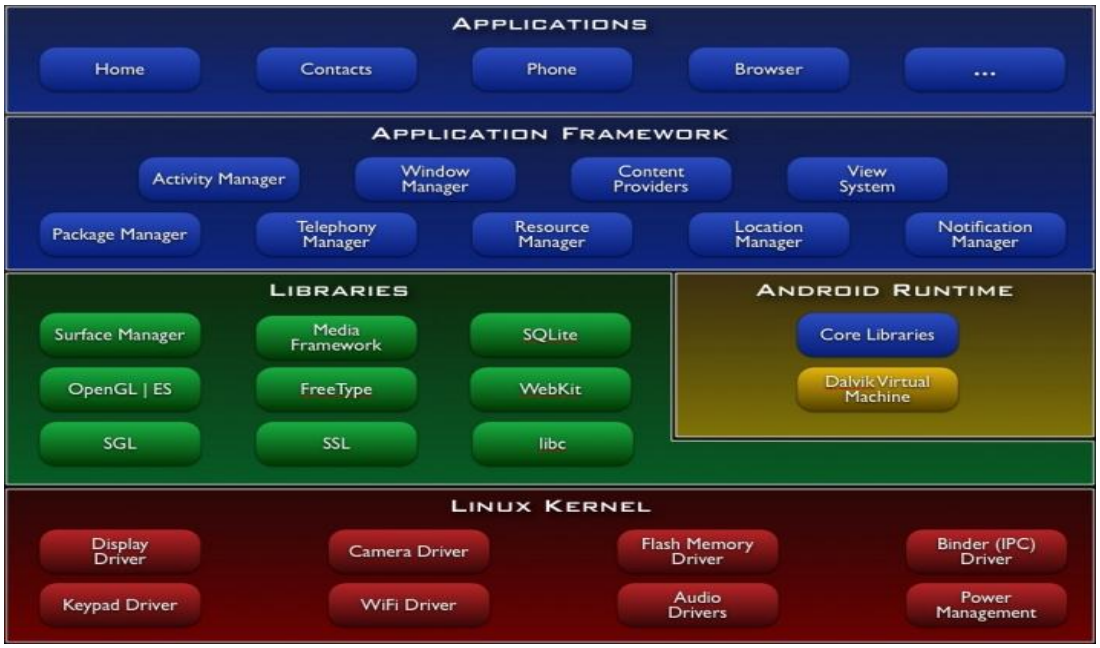
Nejedná se totiž o součást operačního systému. Jedná se o samostatný program, primárně sloužící ke spuštění neboli zavedení operačního systému po spuštění zařízení. Přesněji řečeno k nahrání jádra OS do operační paměti. Většina moderních zavaděčů však zvládá mnoho dalších funkcí. Kromě předání parametrů operačnímu systému, různých testů a dalších funkcí nápomocných při havárii OS patří mezi nejdůležitější funkci mobilních zavaděčů smazání a nahrání nové ROM. Pokud máte zavaděč odemčený, lze se do něj dostat tak, že při zapínání telefonu přidržíte

nějakou kombinaci kláves. Tuto kombinaci má však každý telefon odlišnou. Mezi nejznámější alternativní zavaděče patří například ClockworkMod Recovery (6).



Obrázek 1: Recovery mod Androidu.

Na obrázku pod tímto odstavcem je architektura androidu pěkně znázorněna. Pojďme si tedy jednotlivé pojmy popsat a vysvětlit (6).



Obrázek 2: Pohled na architekturu Androidu.

ROM:

Taktéž se ještě nejedná o součást operačního systému, ale úzce s ním souvisí. Je to zvláštní část paměti zařízení, do které lze zapisovat pouze ve zvláštním režimu. Jedná se totiž o část paměti, ve které je uložen vlastní operační systém. Slovem ROM se také označují vlastní soubory s operačním systémem. Součástí ROM je také takzvaná Radio ROM. To je součást paměti, ve které jsou uloženy informace o operátorovi a základní ovladače hardwaru, především GSM čipu. Bývá zde také uložen SIM Lock (blokování telefonu na jednoho operátora). Další částí ROM je Extended ROM. To jsou různé programy a vlastní úpravy systému od výrobce nebo od operátora. V androidu jsou zde uloženy například Google apps. Posledním důležitým pojmem je CID Lock (Carrier ID). Jinak také VendorLock. Jedná se o mechanismus od výrobce či distributora zařízení, který má zabránit nahrání neoficiálních ROM. CID Lock lze odstranit, uživatel tím však přichází o záruku (6).

Kernel (jádro):

Nyní se konečně dostáváme k první části vlastního operačního systému. Na obrázku je zobrazen úplně dole. Kernel je základem operačního systému a především zajišťuje komunikaci mezi hardwarem a softwarem. Mezi jeho hlavní součást patří Drivery (ovladače), které zařizují právě onu komunikaci. Dále také zajišťuje správu procesů, správu paměti, správu napájení, zajišťuje síťové spojení atd. Android nepoužívá vlastní jádro, ale využívá Linuxové jádro, standardně ve verzi 2.6 (6).

Knihovny (Libraries):

Nativní knihovny androidu jsou napsány v C/C++ a jedná se o základní funkce systému. Surfacemanager se stará o zobrazování aplikací a jejich vrstvení. Open GL a SGL jsou knihovny pro práci s grafikou. Open GL pro 3D grafiku a SGL pro 2D Grafiku. Media Framework slouží k práci s mediálními soubory. Obsahuje například kodeky pro různé formáty audia a videa. SQLite slouží pro ukládání a práci s daty. Webkit je open source vykreslovací jádro pro webový prohlížeč, FreeType se stará o vykreslování písma a SSL se stará o šifrování a zabezpečení přenosu dat. Také zde najdeme základní C knihovny (6).

Android Runtime a Dalvik Virtual Machine:

Tato vrstva slouží primárně pro běh aplikací. Jelikož nejsou aplikace napsány v nativním kódu, ale v Javě, nachází se zde Dalvik Virtual Machine, což je aplikační virtuální stroj, který se stará o převod kódu ve kterém jsou napsané aplikace do nativního kódu. Důvod, proč nebyl zvolen Java VM, ale byl vyvinut vlastní virtuální stroj, je především licenční. Java VM a jeho součásti totiž nejsou open source. Dále se zde nachází standartní Java knihovny (6).

Application Framework:

Vrstva obsahující další knihovny, tentokrát napsané v Javě, které tvoří vlastní systémové API, což je soubor funkcí, které umožňují programátorovi pracovat s prvky operačního systému. Jedná se zejména o přístup ke grafickým prvkům systému (tlačítka atp.), obsahu jiných aplikací (např. kontakty), API pro práci s notifikacemi, atd. Nad touto vrstvou už běží samotné aplikace (6).

2.3 Historie verzí Google Android

Od první verze bylo vydáno několik aktualizací, které opravují chyby a přidávají novou funkcionality. Jednotlivé verze systému jsou označovány podle zákusků (Cupcake, Donut, Eclair, Froyo, Gingerbread) (7).

1.5 (Cupcake) Linuxové jádro 2.6.27 (7).

- 30. Dubna 2009 byla uvolněna aktualizace na Android 1.5 (Cupcake). Tento update přidává několik nových funkcí:
- Možnost nahrávat a sledovat videa z kamery.
- Nahrávání videí na YouTube a fotografií na Picasu přímo z telefonu.
- Nová softwarová klávesnice s automatickým dokončováním slov.
- Bluetooth – podpora A2DP.
- Možnost automaticky připojit Bluetooth headset.
- Nové widgety a složky.
- Animace při přechodu mezi obrazovkami.
- Rozšířena funkce kopírovat a vložit.

1.6 (Donut) Linuxové jádro 2.6.29 (7).

- 15. září 2009 bylo uvolněno 1.6 (Donut).
- Vylepšený Android Market.
- Nové prostředí fotoaparátu, kamery a galerie.
- Galerie umožňuje označit více fotografií k vymazání.
- Aktualizované vyhledávání hlasem.
- QuickSearch Box – umožňuje vyhledávat záložky, historii, kontakty a na webu z domovské obrazovky.
- Podpora pro technologie CDMA/EV-DO, 802.1x, VPN, Gesta a syntéza řeči.
- Podpora pro WVGA rozlišení displeje.
- Vylepšení rychlosti vyhledávání a kamery.

2.0/2.1 (Eclair) Linuxové jádro 2.6.29 (7).

- 26. října 2009 bylo uvolněno 2.0 (Éclair) SDK.
- Optimalizována rychlost hardwaru.
- Podpora pro více velikostí a rozlišení displeje.
- Zdokonalené uživatelské prostředí.
- Nové prostředí prohlížeče a podpora HTML5.
- Nový seznam kontaktů.
- Mapy Google aktualizovány na 3.1.2.
- Podpora pro Microsoft Exchange.
- Podpora přisvětlovací diody.
- Digitální zoom (fotoaparát).
- Vylepšená softwarová klávesnice.
- Podpora pro Bluetooth 2.1.
- Animované tapety na domovské stránce.

2.2 (Froyo) Linuxové jádro 2.6.32 (7).

- 20. května 2010 na konferenci Google I/O byl představen Android 2.2. Přidává nové technologie a funkce uživatelského prostředí. Jedná se o zásadní upgrade, i když z čísla verze to není patrné:
- Možnost instalovat aplikace na paměťovou kartu.
- Adobe vydalo plugin Adobe Flash 10.1. Není integrován do systému, distribuce je řešena přes Android Market nebo přes stránky Adobe.
- Díky JIT (Just-in-time) kompilátoru se podařilo zvýšit rychlost systému na různých benchmarcích 2x až 5x. Dále je vylepšena správa paměti RAM.

- Možnost vytvořit z telefonu WiFihotspot, nebo sdílet internetové připojení přes USB kabel.
- Dva nové režimy telefonu – „car mode“ a „night mode“ (režim v autě a noční režim).
- Více nastavení fotoaparátu a kamery.
- Přidána podpora pro Open GL ES 2.0, vícebarevný trackball, vylepšena podpora pro Exchange, Bluetooth a přidána další vrstva vývojářského API.

2.3/2.4 (Gingerbread), Linuxové Jádro 2.6.35 (7).

- 6. prosince 2010 byla vypuštěna verze Android 2.3 Gingerbread.
- Podpora video formátu WebM pro HTML5 video.
- Podpora pro NearFieldCommunication standard, který dnes podporují některé mobilní telefony.
- Podpora SIP protokolu pro internetovou telefonii.
- Lepší správa prostředků.
- Upravená virtuální klávesnice.
- Zlepšená funkce kopírovat a vložit.
- Podpora více kamer a nových sensorů.
- Nové Google Maps 5 s 3D přístupem (Dostupné přes market pro všechny verze androida).
- Rozšíření podpory nativního kódu.

3.0 (Honeycomb) (7).

- Verze pro tablety. Změny zahrnují:
- Optimalizaci pro velké obrazovky tabletů.
- Spousta nových prvků uživatelského rozhraní.
- Podpora více jádrových procesorů.
- Hardwarová akcelerace pro grafiku.
- Přístup ke Google eBooks.

4.0 (IceCreamSandwich) (7).

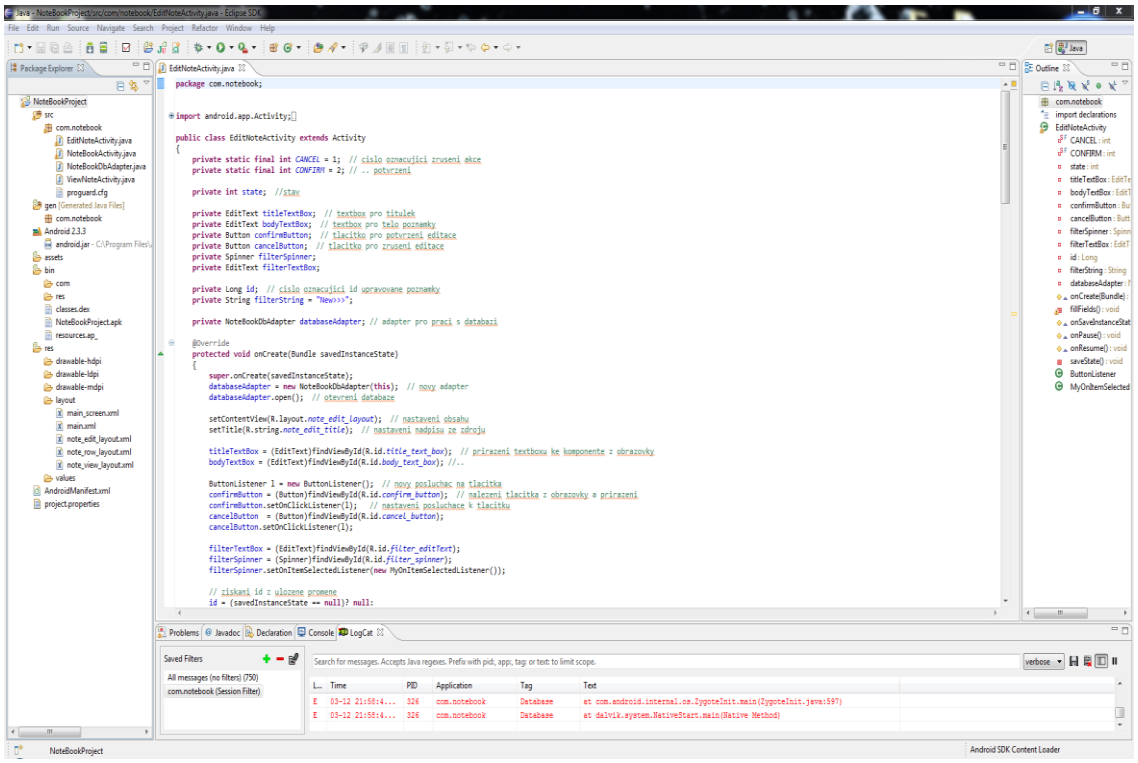
- 19.října 2011 byla představena verze 4.9, která přináší funkcionalitu verze Honeycomb do chytrých telefonů.
- Nové bezpečností funkce.
- Rozpoznávání obličeje.
- Sjednocené kontakty sociálních sítí.

- Offline vyhledávání v emailu.
- Sdílení informací pomocí NFC.

2.4 Android software development kit (SDK)

Aplikace pro operační systém Android jsou vesměs vyvíjeny v jazyce Java. Android má pro vývoj svoje vlastní SDK, které se od klasického liší svým API, tedy aplikačním rozhraním (knihovna procedur a tříd). Android tak má svoje vlastní API přizpůsobené potřebám mobilních zařízení a jejich hardwaru. Tyto nástroje zjednodušují vývoj aplikací. Najdeme zde obrazy systémů, tedy v podstatě základní aplikace telefonu pro volaní, sms, atd. V souboru se také nachází příklady a ukázkové aplikace, což je dobré pro začínající programátory. Součástí Android software development kit (SDK) je implementován Emulátor pro všechny verze androidu. Emulátor je virtuální zařízení, které slouží k testování naprogramované aplikace. Chová se stejně jako mobilní telefon s danou verzí Androidu.

2.5 Vývojové prostředí Eclipse



Obrázek 3: Screenshot vývojového prostředí Eclipse.

Eclipse je open source vývojová platforma, která je pro většinu lidí známa jako vývojové prostředí (IDE) určené pro programování v jazyce Java. Flexibilní návrh této platformy dovoluje rozšířit seznam podporovaných programovacích jazyků za pomoci pluginů, například o C++ nebo PHP. Právě pluginy umožňují toto vývojové prostředí rozšířit například o návrh UML, či zápis HTML nebo XML. Oproti ostatním vývojovým prostředím v Javě, jako například Netbeans, je filozofie Eclipse úzce svázána právě s rozšiřitelností pomocí pluginů. V základní verzi obsahuje Eclipse pouze integrované prostředky pro vývoj standardní Javy jako kompilátor, debugger atd., ale neobsahuje například nástroj pro vizuální návrh grafických uživatelských rozhraní desktopových aplikací nebo aplikační server – všechna taková rozšíření je potřeba dodat formou pluginů. Z tohoto důvodu přímo pod křídly Eclipse vznikly takzvané sub projekty, které zastřešují rozšíření pro jednotlivé oblasti softwarového vývoje v Javě. Tyto sub projekty usnadňují integraci potřebných rozšíření do samotného vývojového prostředí. Eclipse je v současnosti nejpopulárnější IDE pro Javu. Projekt Eclipse (Eclipse 1.0) vznikl uvolněním kódu IBM pod EPL licencí. Hodnota tohoto příspěvku open source se odhaduje na 40 miliónu dolarů. Pro účely tohoto projektu byl vyvinut grafický framework SWT. Výhodou SWT je nativní vzhled aplikací na každé platformě, kde je SWT portován (SWT využívá nativního kódu operačního systému). Naproti tomu konkurenční framework Swing využívá pouze služby JVM, což umožňuje lepší portovatelnost (omezenou pouze dostupností Javy pro danou platformu) (8).

Základní stavební kameny v aplikacích Android jsou komponenty tedy: Aktivity, Dodavatelé obsahu, Služby a záměry, které v odstavci podrobně popíšu:

2.6 Aktivity – (Activity)

Aktivity můžeme chápat jako analogii oken či dialogů aplikace pro stolní počítač. Ačkoliv je možné, aby aktivity neměly uživatelské rozhraní, většina vašeho zdrojového kódů bez uživatelského rozhraní bude zabalená spíše ve formě dodavatelů obsahu služeb (5).

2.7 Dodavatelé obsahu – (Service)

Dodavatelé obsahu zajištují úroveň abstrakce jakýchkoliv dat uložených v zařízení, která jsou přístupná více různými aplikacemi. Vývojový model systému Android vás podporuje v tom, abyste zpřístupnili svá data i jiným aplikacím než pouze

své vlastní. Dosáhnete toho pak právě vytvořením dodavatele obsahu, který vám současně poskytuje úplnou kontrolu nad způsobem přístupu k vašim datům (5).

2.8 Služby – (Content provider)

Aktivity a dodavatelé obsahu jsou entity s krátkou životností a lze je kdykoliv vypnout. Služby jsou oproti tomu navržené tak, pokud je to potřeba, pokračovaly ve své práci nezávisle na jakékoliv aktivitě. Službu můžeme použít k deklaraci aktualizací RSS zdroje nebo k přehrávání hudby, které pokračuje, dokonce i když už byla příslušná ovládací aktivita uzavřená (5).

2.9 Záměry – (Broadcastreciever)

Záměry jsou systémové zprávy, upozorňující aplikace na výskyt různých událostí, změnami hardwarové konfigurace počínaje (například vložení SD karty) přes události související s příchozími daty (například přijetí SMS Zprávy) a událostmi aplikace konče (například její spuštění z hlavního menu zařízení). Na záměry pak můžete nejenom reagovat, ale můžete také vytvářet své vlastní záměry a spouštět jejich prostřednictvím jiné aktivity nebo je využívat k detekci specifických situací (například vyvoláte takový a takový záměr, když se zařízení dostane do vzdálenosti 100 metrů od takové a takové lokace) (5).

3. Porovnání nejstahovanějších manažerských aplikací pro platformu Google Android

Já jsem se zaměřil na aplikace, které manažeři dnes a denně používají. „Ano“ jsou to diáře nebo-li poznámkové bloky. Všechny tyto aplikace nabízí spoustu funkcí a nastavení např. integrovaný kalendář, automatické upozornění na poznámky, podpora widgetu, ochrana heslem, časový alarm, online zálohování, barevné znázornění poznámek atd. Otázka zní: využijeme opravdu všechny funkce a co hardwarová náročnost na systém?

3.1 Analýza aplikací (Android Market)

Co je to Android Market? Je to služba provozovaná od Googlu. Vlastníci mobilního zařízení s operačním systémem Android si mohou díky této službě stáhnout na mobilní zařízení jakoukoliv aplikaci, pokud tím Android Market disponuje. Aplikace mohou být placené i volně stažitelné. V Marketu najdeme tisíce a tisíce aplikací, které mají široký rozsah využití.

Nyní vybrané aplikace podrobím zkušebním testům. Pro srovnání jsem si vybral tři aplikace volně šiřitelné, které jsou mezi uživateli nejoblíbenější. Ke stahování aplikací existují ještě neoficiální servery, ale těmi se nebudu zabývat, z důvodů, že většina uživatelů zde nemá přístup a měření by následně nebylo objektivní. Ke srovnání budu využívat pouze server Android Market.

Abych mohl vyhodnotit aplikace, musel jsem si zvolit kritéria, podle kterých určím, jak má vypadat návrh mé aplikace.

Kritéria:

Velikost aplikace - Je důležité, aby telefon měl dostatek paměti na další procesy a aplikace, protože při nedostatku paměti telefon ztrácí výkonnost.

Hardwarové vytížení systému - Pokud je aplikace náročná nebo byla aplikovaná špatná syntaxe zdrojového kódu, aplikace se mohou zamrzat nebo dokonce ovlivnit celý běh telefonu.

Jednoduchost aplikace – Čím jednodušší aplikace tím lépe se v ní orientujeme, ale nemusí to být podmínkou.

Rychlost a stabilita aplikace – Problematika s rychlostí se spíše bude týkat starších telefonů Google Android. Stabilita aplikace zaleží na použitém hardwaru, kterým zařízení disponují a také jak je zdrojový kód náročný na překlad.

Filtrování poznámek – Filtr slouží, aby uživatel mohl jednoduše přistupovat ke svým datům popřípadě slučovat vybrané poznámky k sobě.

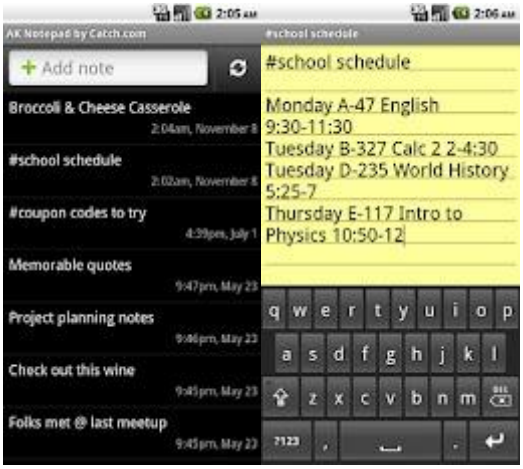
1. ColorNote – nejlépe hodnoceným ze (112 697) stáhnutím.

Obrázek 4: Screenshot aplikace ColorNote převzaté z Android Marketu.

2. JorteCalendar - nejlépe hodnoceným ze (78 476) stáhnutím.

Obrazek 5: Screenshot aplikace JorteCalendar převzaté z Android Marketu.

3. AK Notepad - nejlépe hodnoceným ze (50949) stáhnutím.



Obrazek 6: Screenshot aplikace AK Notepad převzaté z Android Marketu.

Testoval jsem za pomocí vlastního telefonu: LG OptimusOne, Android 2.3.3 Gingerbread, RAM 512 MB, vnitřní paměť 170 MB, procesor 600 MHz Qualcomm.

Analyzoval jsem za pomocí aplikace: CoolTool 3.1.6.

Stabilní systém hodnot telefonu: CPU 9% při 245 MHz, 262 MB RAM.

	ColorNote	**AK Notepad**	**JorteCalendar**
Velikost aplikace	642 kb	468 kb	2,9 Mb
Hardwarové vytížení systému	*CPU 34% 245 MHz, 6 MB RAM	*CPU 37% 245 MHz, 5 MB RAM	*CPU 54% 480 MHz, 9 MB RAM
Jednoduchost aplikace	složitý	jednoduchý	složitý
Rychlost a stabilita aplikace	Rychlá reakce stabilní	Rychlá reakce stabilní	Zasekává se, aplikace občas padá
Filtrování poznámek	ano	ano	ano

Tabulka 1: Srovnání aplikací vybraných aplikací.

*CPU – procesor, MB RAM – operační paměť, MHz – frekvence procesoru,

Test probíhal vyhodnocením součtu zpuštěných aplikací (IDLE), aplikací v zátěži (BURN) a psaní poznámek. Výsledkem je průměr naměřených hodnot. Samozřejmě, že tato čísla se budou lišit, když použijeme jiný telefon nebo jinou verzi systému.

3.2 Vyhodnocení stávajících aplikací

ColorNote - je aplikace sloužící převážně k tomu, abychom měli poznámky na ploše telefonu neboli tzv. widget. Hodně krát jsem poznámku na ploše omylem přesunul jinam, což může některým uživatelům vadit. Aplikace nabízí: kalendář, archiv poznámek, koš, změna pozadí – theme a vyhledávání. Zajímavostí je, že můžeme přiřadit poznámce barvu, datum a čas.

AK Notepad - je jednoduchý poznámkový blok, kde můžeme najít například synchronizaci s sms zprávami a emaily, změna pozadí – theme, zámek poznámek apod. Za zmínku stojí říci také to, že dokáže stáhnout poznámky i ze serveru.

JorteCalendar - je poměrně složitá aplikace. Jedná se převážně o kalendář než o poznámkový blok. Nabízí spoustu nastavení a možností orientace v poznámkách, je nepřehledná a lehko se ztratíte v menu. Aplikace je nestabilní dvakrát se mi ukončila. Dále aplikace nabízí: nastavení písma, úkoly a připomínky, několik druhů kalendářů, alarm atd.

Dle následujícího porovnání vidíme, že každá aplikace je dobrá v něčem jiném a vždy záleží na segmentu využití softwaru. Každý uživatel se může rozhodnout sám, co vlastě od softwaru očekává. Všechny aplikace nabízí hezké vizuální prostředí a spoustu doplňkových věcí navíc, ale podle mého názoru to jsou zbytečnosti, které zřídka nebo nikdy nevyužijeme.

3.3 Návrh aplikace

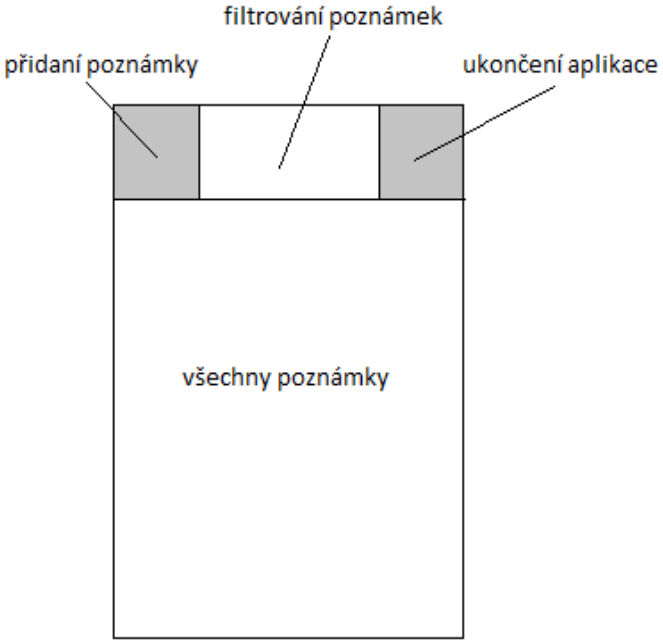
V kapitole jsem analyzoval vybrané aplikace podle vybraných kritérií, kde jsem narazil na problematiku stávajících poznámkových bloků. Zjistil jsem, že aplikace jsou náročné na systém telefonu a v jisté míře zbytečně složité na obsluhu. Dalším nedostatkem je kompatibilita se staršími verzemi systému Google Aneroid, což znemožňuje používání u některých telefonů. Z tohoto důvodu jsem se rozhodl navrhnout novou aplikaci, která musí být jednoduchá a nijak nenáročná na systém

telefonu a bude se všemi verzemi Google Android kompatibilní. Myslím si, že nejcennější pro manažery je čas a rozvržení vytyčených úkolů. Výstupem je zjednodušení práce manažerům s vedením poznámek tak, aby nemuseli zdlouhavě nastavovat a vypisovat údaje, které nejsou tak důležité při jejich práci a mohli se soustředit na jiné aktivity. Sami dobře víme, že nejsme roboti a poznačit si myšlenku v dané chvíli, než jí zapomeneme je k nezaplacení.

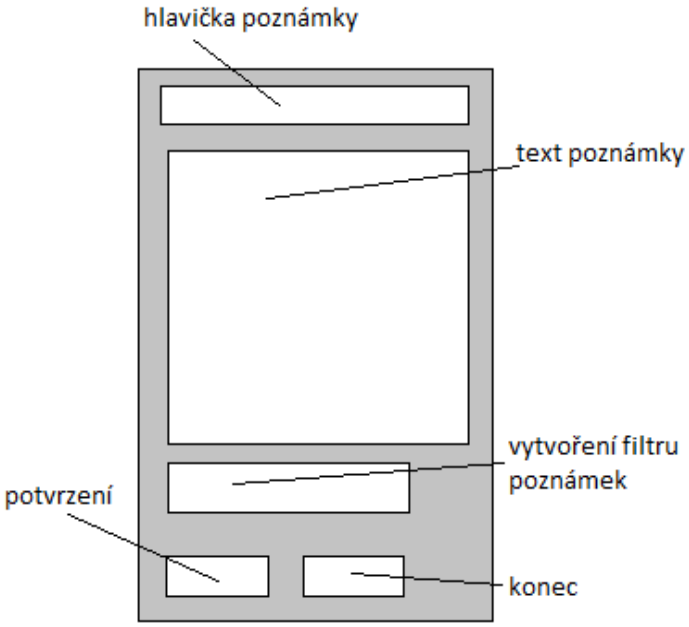
3.4 SWOT analýza aplikace

Silné stránky	Příležitosti
Jednoduchost **Rychlost** **Ovladatelnost** **Přehlednost** **Stabilita** **Hardwarové nároky** **Aplikace nejen pro manažery** **Spustitelné na všech verzích Androida**	**Aktualizace** **Nové verze Androida** **Uchycení Aplikace na Android Marketu**
Slabé stránky	Ohrožení
Jednoduchost – někomu nemusí vyhovovat. **Vzhled aplikace**	**Podobná aplikace** **Konkurence**

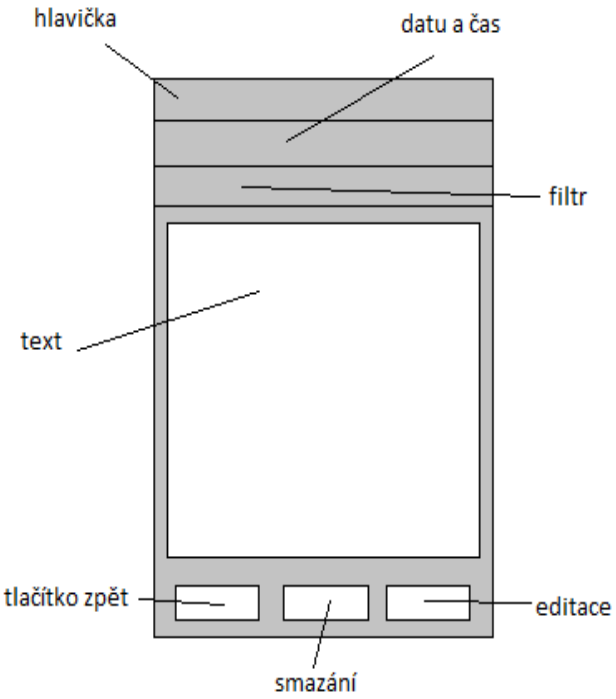
Abych mohl začít programovat aplikaci, musel jsem si navrhnout hrubý koncept poznámkového bloku, jak by asi měl vypadat. Za cíl jsem si stanovil, aby aplikace vypadala co nejjednodušší z důvodu přehlednosti a orientace v poznámkách.



Obrázek 7: Návrh úvodní obrazovka.

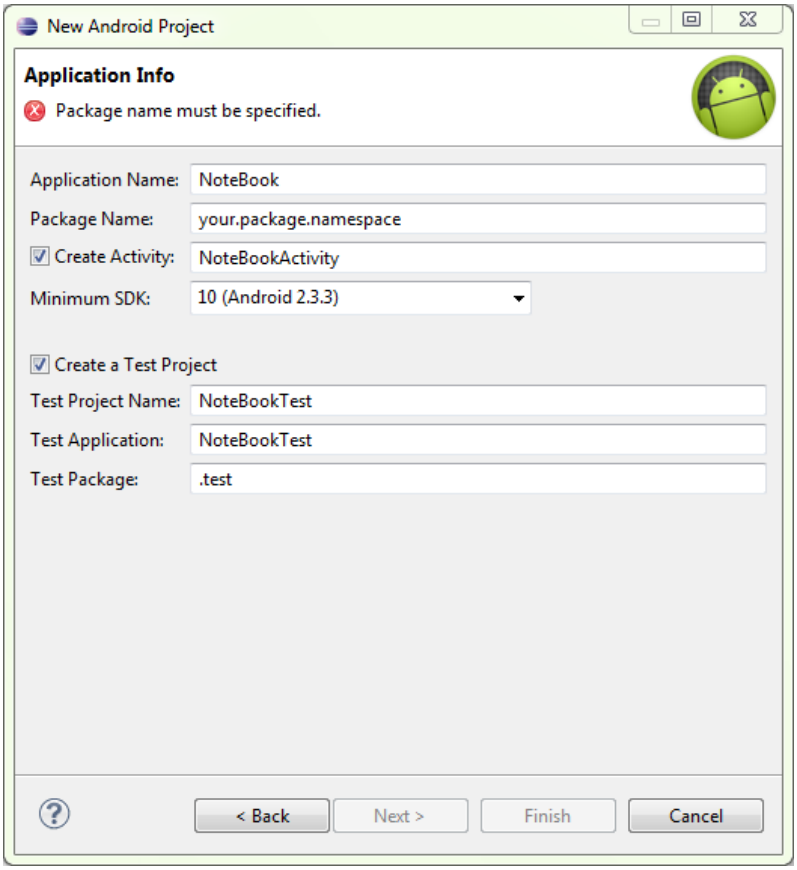


Obrázek 8: Návrh vytvoření poznámky.



Obrázek 9: Návrh prohlížení poznámky.

4. Vývoj aplikace a založení Android projektu



Obrázek 10: Založení Android projektu.

Pro vývoj aplikace budeme potřebovat Java development KIT známý taky jako (JDK) a vývojové prostředí Eclipse, kde bylo nutné nainstalovat Android software development kit (SDK) a zásuvný modul Android Development Tools (ADT). Po nainstalování všech potřebných pluginů do Eclipse se objeví v programu nová volba menu pro založení Android projektu. Nyní můžeme založit Android projekt a začít programovat aplikaci. Klikneme na vytvořit nový Android projekt, a zobrazí se nám dialog, kde musíme vyplnit položky: název projektu, místo kde se aplikace bude tvořit a ukládat, nakonec musíme vybrat verzi SDK neboli verzi Androidu (10).

4.1 Přehled souborů v projektu

Ve složce src jsou uložené všechny třídy, které jsme vytvořili při programování, nachází se zde čtyři třídy EditNoteActivity.java, NoteBookActivity.java,

NoteBookDbAdapter.java, ViewNoteActivity.java. Tyto třídy tvoří jádro celé aplikace. Dále je zde složka gen, ve které se nachází všechny kódy generované vývojovým prostředím. V našem případě obsahuje jeden soubor R.java, který obsahuje všechny číselné odkazy k jednotlivým prvkům grafického uživatelského rozhraní. Projekt má importovanou knihovnu Android.jar, ve které je celé API potřebné pro vývoj androidních aplikací. Ve složce res se nacházejí všechny resources, tedy zdroje. Tím je například ikona aplikace, uložená ve všech složkách drawable, tedy hdpi, ldpi a mdpi. Ve složce layout se nacházejí všechny layouty, tedy rozvržení jednotlivých aktivit, které se budou zobrazovat na displeji telefonu při práci s aplikací. Všechny soubory v této složce jsou strukturované pomocí jazyka xml. Vývojové prostředí umožňuje zobrazit přímo grafický vzhled jednotlivých layoutů. Můžeme tak navrhnout celou aplikaci.

Ve složce values se nacházejí všechny hodnoty, které chceme mít uložené přímo v aplikaci (tzv. natvrdo). Tím máme oddělena data od kódů a při změně hodnot, není potřeba měnit kód. V našem případě se zde nachází soubor strings.xml, ve kterém jsou uloženy všechny možné textové řetězce, které jsou viditelné v aplikaci. Například různé nadpisy apod. Posledním důležitým souborem, který se zde nachází je AndroidManifest.xml, ve kterém se definují různé atributy pro celou aplikaci, například seznam povolených aktivit, nastavení ikony aplikace, název aplikace, různá oprávnění a nastavení hlavní aktivity, která se spustí při otevření aplikace.

4.2 Programátorská část

Při vývoji aplikace bylo čerpáno z následujících zdrojů: (11), (12), (13), (14), (15). Bylo využito implementovaných tříd a jejich funkcí z Android API, které jsou popsány ve zdroji (16).

4.2.1 Třída NoteBookActivity

Tato třída tvoří hlavní jádro aplikace. Dědí (extends) se ze třídy ListActivity, která dědí ze třídy Activity. Activity je aplikační komponenta, která poskytuje výstupní obraz spuštěné aplikace. ListActivity je pak rozšířena o možnost skládat do této aktivity seznam prvků, například dalších menších buněk, které jsou svázány s nějakými daty například z Cursoru. Toho se aplikace využívá pro seřazení seznamu záznamů jednotlivých Poznámek. Rozvržení jednotlivých funkčních prvků v Activitě pak definuje Layout, což je soubor, který je strukturován pomocí jazyka xml. V tomto případě je

hlavní obrazovka této aktivity strukturována pomocí souboru *main_screen.xml*. O rozvržení prvků se starají `LinearLayout`, které řadí jednotlivé prvky lineárně za sebe vertikálně či horizontálně. Hlavní panel pro ovládání aplikace se nachází nahoře a je řazen horizontálně. Nachází se v něm tlačítko *Button* +, které slouží k vytvoření nové poznámky, dále pak *Spinner*, který nabízí k výběru filtrování podle určitých skupin, do kterých přiřazujeme jednotlivé poznámky, a nakonec *Button* Exit, pro opuštění aplikace. Další důležitou komponentou Layoutu je `ListView`, sloužící pro zobrazení seznamu poznámek.

Ve třídě NoteBookActivity jsou jednotlivé prvky jako Button a Spinner definována jako třídní proměnné, aby bylo možno s nimi pracovat v rámci celé třídy. Jsou to tedy `newNoteButton`, `filterSpinner`, `exitButton`. Pro kýženou funkci Activity bylo nutné přepsat (@Override) několik metod.

Nejdůležitější metodou pro samotný běh a zobrazení uživatelského rozhraní je metoda `public void onCreate(Bundle savedInstanceState).` Tato metoda je vyvolána při vytváření Activity, tedy v tomto případě při spuštění celé aplikace uživatelem. V této metodě se nejdříve zavolá nadtřída, ze které se dědí, pak se nastaví pohled na obsah `setContentView(R.layout.main_screen);` dále se vytvoří databázový adaptér, který slouží pro práci s databází poznámek a otevře se. Dále se inicializují jednotlivé ovládací prvky (tlačítka) a naleznou se jejich pohledy v zobrazovaném Layoutu. To znamená, že se nalezne v main_screen, kde se má tlačítko nacházet, jak má vypadat (barva, písmo) atd. Následně se ovládacím prvkům přiřadí tzv. Listenery. Listener, tedy posluchač, čeká dokud se s ovládacím prvkem neprovede nějaká akce. Pokud se provede (stiskne tlačítko), listener vyvolá metodu která se nachází v listeneru. Tlačítkům je přiřazen `OnClickListener` a Spinneru `OnItemSelectedListener`.

Následně se zavolá metoda refreshData(); , která slouží pro obnovení dat v aktivitě, tedy zobrazované poznámky. Nejdříve se načtou z databáze do Cursoru c všechny poznámky - Cursor c = `databaseAdapter.selectAllNotes();` To platí pro filtrování, kdy je zadáno že se mají zobrazovat všechny poznámky (All). Pokud však je vybráno jiné filtrování (podmínka if), načte se z databáze do Cursoru vyfiltrované poznámky pomocí: c = `databaseAdapter.selectNotesByFilter(filterString);` Aktivita začne spravovat Cursor. Vytvoříme `SimpleCursorAdapter notes = new SimpleCursorAdapter(this, R.layout.note_row_layout, c, from, to);`

Tento adaptér slouží k načtení dat z Cursoru *c*, přiřazení layoutu *note_row_layout* každému řádku a to pomocí pole *from* (z), kde je definováno, které atributy se transformují na jednotlivé kolonky layoutu a to pomocí pole *to* (do). To znamená, že například titulek poznámky se má zobrazit v řádku na místě definované note_row_layout, do `TextView` s `id/row1`. Nakonec se tento seznam transformovaných poznámek na pohledy pomocí jednotlivých řádků nastaví jako `setListAdapter(notes);`

Po vykonání metody refreshData se spustí další metoda `refreshSpinner();` která obnoví seznam jednotlivých filtrovacích skupin, které jsme sami vytvořili. Opět se vytvoří Cursor, do kterého načteme z databáze pomocí `Cursor cs = databaseAdapter.selectFilters();` všechny filtry. Dále se vytvoří textové pole *filters,* do kterého se pomocí cyklu *for*, načtou všechny názvy filtrů z Cursoru. Dále se vytvoří adaptér, kterému nastavíme seznam filtrů a layout a tento adaptér přiřadíme *filterSpinneru.* To má výsledek ten, že pokud klikneme na ovládací prvek filterSpinner, tak se nám zobrazí seznam všech filtrů, podle kterých se dají filtrovat poznámky. Standardně je vždy zobrazován filterAll, kdy se zobrazují všechny poznámky, bez ohledu na filter. Pokud vytvoříme poznámku s novým filtrem, vymažeme všechny poznámky s nějakým filtrem, či upravíme název filtru v poznámce, to vše by se mělo projevit v tomto Spinneru, pokud je zavolána metoda refreshSpinner.

Další metodou, které jsme přepsali je například `@Override public boolean onCreateOptionsMenu(Menu menu)`, ta se spouští také při spuštění samotné aplikace. Zde pouze přidáváme do menu nové tlačítko pro přidání nové poznámky.

`Public void onCreateContextMenu(ContextMenu menu, View v, ContextMenu InfomenuInfo)` Tato metoda se sama spouští při spuštění aplikace, slouží k vytvoření contextMenu. Zde pouze přidáváme do context menu tlačítko pro vymazání vybrané poznámky.

`@Override public boolean onContextItemSelected(MenuItem item)`

Tato metoda se spustí (pomocí vnitřních posluchačů) pokud vybereme nějakou položku z contextMenu nad vybranou poznámkou. Zde context menu slouží pouze pro mazání poznámek, takže pokud je vybraná poznámka označena a je vyvoláno contextMenu a zvolí se deletenote, pak se z přiložených informací v item zjistí identifikační číslo poznámky, které určuje jednoznačně poznámku v tabulce a pak pomocí této informace vymaže poznámku z databáze

`databaseAdapter.deleteNote(info.id);` a obnoví zobrazované poznámky, tedy vymazaná poznámka zmizí ze seznamu.

Metoda `public boolean onMenuItemSelected (int featureId, MenuItem item)` je vyvolána v okamžiku, kdy vybereme položku v menu. Zde je jen dříve zmiňované vytvoření nové poznámky. Dojde tedy k tomu, že se vyvolá následně metoda `createNewNote()` sloužící k vytvoření nové poznámky, k této metodě více níže.

Stlačením jakéhokoliv tlačítka v aplikaci, které má přidělena třída `ButtonListener` se spustí v této třídě, která musí implementovat (implements) rozhranní (interface) `OnClickListener.`

Třída `ButtonListener`, aby bylo možné s ní pracovat jako s listenerem, který naslouchá všem tlačítkům, kterým byl přidělen, musí implementovat (implements) rozhranní (interface) a to `OnClickListener`. Stlačením jakéhokoliv tlačítka, kterému je posluchač přidělen, se spustí metoda `public void onClick(View v)` v této třídě (`ButtonListener`). Předávaný parametr pohledu představuje objekt, ze kterého byla tato událost vyvolána. Přetypováním na tlačítko pak zjistíme, které tlačítko bylo zmáčknuto. To se tedy provede tak, že pohled `v`, přetypujeme na tlačítko – `(Button) v`. Porovnáním (`==`) s vlastním objektem tlačítka zjistíme, jestli se jedná o dané tlačítko, pomocí příkazu if. Pokud je stlačeno tlačítko newNoteButton, vyvolá se metoda `createNewNote()`. Pokud je stlačeno tlačítko exitButton, pak se zavolá metoda `finish()`, která bezpečně ukončí celou aplikaci.

Třída `MyOnItemSelectedListener`, aby bylo možné s ní pracovat jako s listenerem, který naslouchá v našem případě Spinneru, musí implementovat rozhraní `OnItemSelectedListener`.

Pokud se změní vybraná položka Spinneru, kterému je naslouchán, vyvolá se v dané třídě metoda:

`Public void onItemSelected(AdapterView<?> parent, View view, int pos, long id)` ve které změníme pouze parametr filterString na hodnotu vybrané položky a následně se obnoví seznam zobrazovaných poznámek podle vybraného filtru.

Metoda `private void createNewNote()` slouží k vytvoření nové poznámky. Tato metoda je volána z menu a pomocí tlačítka (+). V této metodě se

vytvoří nový cíl Intent *i* jako nová aktivita **`EditNoteActivity`**. Tato nová aktivita se pak spustí a přejde se k ní, zatímco současná aktivita se přeruší a čeká se na výsledek, který se vrátí po ukončení nové aktivity.

Metoda `@Override protected void onListItemClick(ListView l, View v, int position, long id).`

Se vyvolá stlačením položky ze seznamu, tedy poznámky. V této metodě se vytvoří nový cíl **`ViewNoteActivity`**, sloužící pro zobrazení dané poznámky, přidá se extra informace o identifikačním čísle vybrané poznámky a vyvolá se nová aktivita. Tím se tedy přejde k novému oknu nové aktivity.

4.2.2 Třída EditNoteActivity

Tato třída dědí (extends - rozšiřuje) ze třídy *Aktivity*. To znamená, že tato třída se také stává aktivitou. Tato aktivita slouží pro vytvoření úplně nové poznámky (vyvoláním nové aktivity z aktivity NoteBookActivity v metodě **`createNewNote`**), či editování již vytvořené poznámky. Při vytvoření této nové aktivity se automaticky vyvolá přepsaná (@override) metoda z nadtřídy Aktivity:

```
protected void onCreate(Bundle savedInstanceState)
```

V této metodě se nejdříve vytvoří nový databázový adaptér **`databaseAdaptera`** a následně se otevře. (Více o třídě **`NoteBookDbAdapter`** ze které je vytvořená instance databaseAdapter bude popsáno níže). Následně se nastaví pohled na tuto aktivitu. K tomu slouží Layout ***`note_edit_layout`*** ve kterém je definován vzhled této aktivity. V této aktivitě se nachází ***`title_text_box`*** sloužící pro titulek vytvářené/editované poznámky. Dále ***`body_text_box`*** slouží pro text těla poznámky. ***`Confirm_button`*** je tlačítko, které slouží k potvrzení vytváření/editace. Po jeho stlačení se celá upravená poznámka uloží do databáze, stávající aktivita se uloží a vrátí se k předešlé aktivitě NoteBookActivity. ***`Cancel_button`*** slouží pro zrušení úpravy poznámky, tedy změny se neprojeví a neuloží se nic do databáze, daná aktivita se ukončí a vrátí se k předešlé aktivitě NoteBookActivity. ***`Filter_editText`*** slouží pro zadání názvu filtrovací skupiny, podle které se pak může seznam poznámek omezit pouze na danou skupinu. ***`Filter_spinner`*** slouží pro vybrání již existujících skupin, pod kterými jsou uloženy ostatní poznámky (17).

Všechny tyto funkční komponenty se v metodě onCreate naleznou pomocí metody findViewById a vytáhnou do příslušných proměnných. Tlačítkům se přiřadí posluchače (listenery) ze třídy ButtonListener. Komponentě filterSpinner se přiřadí MyOnItemSelectedListener.

Následně se získá ***id*** upravované poznámky z extra informací předaných z předchozí komponenty v proměnné savedInstanceState. Pokud tato proměnná je prázdná porovnání (**savedInstanceState == null**), pak se do proměnné id uloží hodnota null (prázdná hodnota). Pokud však savedInstanceState obsahuje nějaké informace, získá se hodnota id z (**Long**) **savedInstanceState.getSerializable (NoteBookDbAdapter.ID);**

To znamená, že pokud při spuštění této aktivity se nepředají žádné extra informace, jedná se o vytváření úplně nové poznámky. Pokud však přidané informace se nacházejí, jde o editaci již vytvořené poznámky. Tato poznámka je identifikována pomocí proměnné id v tabulce databáze.

Nakonec se v metodě onCreate zavolá metoda fillFields().

Private void fillFields() - Tato metoda plní textová pole, pokud se jedná o editaci poznámky.

Pokud tedy id poznámky není prázdné – if(id!= null), vytvoří se nový Cursor, do kterého se předá z databáze pomocí metody selectNote a parametru id, poznámka, která se má editovat. Následně se nastaví text této poznámky do textových polí. Také se nastaví textový řetězec filterFF na hodnotu filtrovací skupiny (18).

V dalším kroku se vyberou všechny filtrovací skupiny, které se nachází v tabulce poznámek. A to z toho důvodu, abychom si mohli vybrat jednu ze skupin, pokud nechceme zadávat vlastní novou skupinu. To se provede načtením všech těchto skupin pomocí databázového adaptéru, který je napojen na databázi poznámek metodou ***selectFilters()***. Nejdříve se tento kursor, který v sobě uchovává všechny filtry nastaví na začátek (metoda moveToFirst()). Vytvoříme se pole textových řetězců **String[] filters = new String[cs.getCount()+1];** které neinicializujeme na délku hodnoty počtu prvků, tedy počtu filtrů v kursoru plus jedna. Je to z toho důvodu, že bychom chtěli vytvořit i vlastní filtrovací skupinu. Proto se tedy na první místo (index 0), v tomto poli uloží text „New>>>“. To znamená, že pokud následně vybereme ve Spinneru tuto hodnotu, tak nechceme přiřadit vytvářenou/editovanou poznámku, do žádné z již vytvořených skupin, ale do vlastní

skupiny. Pak je nutné tedy napsat název nové skupiny do příslušného textového pole, nacházejícím se vedle Spinneru.

Teď přichází na řadu načíst ostatní, již vytvořené filtry z kursoru. To se provede pomocí cyklu `for(int i=1;i<cs.getCount()+1;i++)` kde i = 1 znamená že, hodnotu z kursoru budeme ukládat na od indexu jedna, tedy na druhou. Pozici v poli ***filters*** (první místo má index 0). i<cs.getCount() + 1, znamená, že index bude nabývat do hodnoty počtů filtrů v kursoru plus jedna. i++ pak znamená, že index se bude postupně v každém cyklu přičítat. Tím se tedy úsek kódu mezi složenými závorkami `({})` projde tolikrát, kolik se nachází v kursoru filtrů. V tomto úseku se vždy do pole filtrů uloží aktuální hodnota z kursoru. Dokud kursor není na konci, pak se kursor posune na další prvek, který se v něm nachází:

```
if(!cs.isLast())

{

      cs.moveToNext();

}
```

Mezitím se porovnává, jestli aktuální filtr neodpovídá filtru, který je uložený v poznámce, kterou zrovna editujeme.

```
if(filterFF.equals(filters[i]))

{

      index = i;

}
```

Pokud ano, zapamatujeme si tento index v proměnné ***index***. Dále vytvoříme adaptér, který slouží pro připojení pole filtrů (filters) a jeho jednotlivých položek na jednotlivé layout položky, které se pak zobrazí ve Spinneru. Nakonec se filterSpinner naváže na vytvořený adaptér a tím se tedy zobrazí všechny položky filtrů v tomto Spinneru.

Pokud upravujeme poznámky a tedy nevytváříme novou, pak id se nerovná prázdné hodnotě a tedy můžeme nastavit tomuto filterSpinner ***index*** vybraného filtru. Pokud však filtrovací textový řetězec filterString je roven „New>>>“ (vytváříme novou poznámku a chceme vytvořit novou skupinu filtru), pak textové pole nastavíme, aby

bylo viditelné (View.VISIBLE) a mohli jsme do něj zadat nový název skupiny a tím obnovíme tento textBox. Ale jinak, pokud je filterString jiná hodnota (else) pak se filterTextBox nastaví na viditelnost hodnoty View.GONE a tedy tento textBox není viditelný a nemůžeme do něj nic zadat.

Přepsaná metoda **`protected void onSaveInstanceState(Bundle outState)`** se vyvolá sama, pokud aplikace chce z nějakého důvodu uložit svůj stav a tím třeba přerušit současnou Aktivitu. Něco podobného je i metoda **`protected void onPause()`**, která se vyvolá při nějakém přerušení aplikace. Z toho důvodu se v těchto metodách zavolá metoda ***`saveState()`***, která uloží stav naší aktivity (vyplněná textová pole apod.)

V této metodě se zjistí, jestli je daný stav potvrzený, to znamená, že se chce uložit stav po zmáčknutí tlačítka Confirm, pak vyvoláme hodnoty, které jsou vyplněny v textboxech, vezmeme filtrovací text a to buď nově vytvořený z filterTextBox, či již existující, uložený ve filterString, vybraný spinnerem z databáze. Dále zjistíme, jestli je id == null, tedy pokud je prázdné vytváříme novou poznámku.

Pak tedy vložíme pomocí databázového adaptéru metodou createNewNote novou poznámku a pokud se ji podaří uložit (metoda vrátí do proměnné idTmp hodnotu větší jako 0) pak tato hodnota odpovídá id nové poznámky. Pokud však upravujeme již vytvořenou poznámku, pak pomocí metody na databaseAdapter.updateNote, upravíme tuto již dříve vytvořenou poznámku.

4.2.3 Listenery

Třída ***`ButtonListener`*** implementuje rozhranní OnClickListener, z toho důvodu, aby s ní bylo možné pracovat. Instanci této třídy jsme již dříve přiřadili tlačítkům Cancel a Confirm. Pokud se tedy stlačí jedno z těchto tlačítek, vyvolá se metoda **`public void onClick(View v)`** z této třídy. Předávaný parametr View v představuje přímo tlačítko, které bylo stlačeno. To znamená, že můžeme rozlišit, které tlačítko bylo stlačeno i když byl tento listener přiřazen oběma tlačítkům. Pokud se jedná o tlačítko Confirm a textové pole pro titulek je vyplněné, pak se nastaví výsledek celé aktivity, že je ok, dále nastavíme stav na confirm a metodou finish() dokončíme stávající aktivitu. Tím se tedy vyvolá uložení stavu, tedy metoda onSavedInstance a tím se tedy uloží data v textboxech do databáze, současná aktivita se ukončí a přejde se k předchozí aktivitě NoteBookActivity. Pokud se stlačí tlačítko Cancel,

nastaví se výsledek také na ok, ale stav se nastaví na Cancel, a opět metoda finish() spustí přetíženou metodu onSavedInstanceState. Tím se tedy neuloží žádná data do databáze. Aktivita se ukončí a opět se přejde k předchozí aktivitě NoteBookActivity.

Třída **`MyOnItemSelectedListener`** implementuje rozhraní OnItemSelectedListener. Instance této třídy byla již dříve předána Spinneru pro výběr filtrovací skupiny. Tato třída implementuje dvě metody. Metoda **`public void onItemSelected(AdapterView<?> parent, View view, int pos, long id)`** se volá, pokud se vybere nějaký prvek ze seznamu prvků. Pokud je tedy něco vybráno, nastaví se textový řetězec na filterString na hodnotu vybrané položky. Pokud je filtrovací řetězec nastaven na „New>>>“, pak se jedná o to, že se bude brát v potaz text v textovém poli, nikoli hodnota z tohoto Spinneru. Pak se tedy zviditelní tento textbox, pokud tedy byl neviditelný. Pokud však je vybrána jiný položka, pak se bude jednat o výběr jednoho prvku ze seznamu a není nutné, aby filterTextBox byl viditelný, proto se jeho viditelnost nastaví na hodnotu View.GONE, což znamená neviditelnost. Druhá implementovaná metoda **`public void onNothingSelected(AdapterView<?> parent)`** se zavolá, pokud nebyla vybrána žádná položka ze seznamu. V našem případě není nutné nic provádět a tato metoda nemá pro nás žádný význam. Avšak OnItemSelectedListener tuto metodu požaduje, aby byla implementována.

4.2.4 Třída/Aktivita ViewNoteAcitivy

Tato aktivita slouží pro pouhé zobrazení celé poznámky a neumožňuje úpravu poznámky. Avšak je možné z této aktivity přejít tlačítkem k editaci v aktivitě EditNoteAktivity. Vytvořením této aktivity v aktivitě NoteBookAdapter se zavolá přetížená metoda **`protected void onCreate(Bundle savedInstanceState).`**

V této metodě se vytvoří a otevře nový databázový adaptér. Nastaví se vzhled z layoutu note_view_layout. Následně se vytáhnou ovládací prvky, jako jsou textová pole, Spinner a tlačítka do proměnných, aby bylo možné s nimi pracovat. Přiřadí se tlačítkové posluchače tlačítkům. Následně se získá ***`id`*** poznámky, která se má zobrazit. Nakonec se zavolá metoda fillFields(), která naplní všechna textová pole. V této metodě, pokud je nastavené ***`id`*** na nějakou hodnotu, pak se vytvoří nový kursor a předá se do něj data o poznámce, kterou chceme zobrazit. To se provede zavoláním metodou selectNote(id), kde se pomocí identifikačního čísla předaného z aktivity vybere z databáze, hledaná poznámka. Data o této poznámce se vytáhnou do

jednotlivých textových polí a zobrazí se na obrazovce. Třída ButtonListener implementuje jednu metodu z rozhraní OnClickListener a to **`public void onClick(View v)`**(19).

Která se vyvolá po stlačení tlačítka, kterému byl tento Listener předán. V této metodě se pomocí podmínek if rozhodne o jaké tlačítko se jedná. Pokud bylo stlačeno tlačítko backButton, pak se nastaví výsledek na ok a ukončí se tato aktivita a dojde k návratu do předchozí aktivity. Pokud se stlačí tlačítko editButton, pak se vyvolá metoda **`public void EditButtonClicked()`**. Ve které se vytvoří nový cíl, do kterého přidáme extra informaci o identifikačním čísle editované poznámky a metodou **`startActivityForResult(i, 1);`** se přeruší aktuální aktivita a vyvolá se nová aktivita EditNoteActivity. Po následné editaci poznámky v edit aktivitě a ukončení této aktivity se navrátí k aktivitě metodou **`protected void onActivityResult(int requestCode, int resultCode, Int entintent)`** ve které pouze obnovíme zobrazovaná data v textových polích metodou fillFields(). Stlačením tlačítka deleteButton, se pomocí databázového adaptéru zavolá metoda deleteNote(id), ve které se pomocí předaného id vybrání poznámka vymaže z databáze. Protože tato poznámka je již vymazána z databáze, není nutné zobrazovat dále data a může se aktivita ukončit.

4.2.5 Třída NoteBookDbAdapter

Představuje pomocný objekt pro práci s databází. Jako třídní proměnné jsou zde uložené určené pro vytvoření databáze a samotné tabulky, do které se ukládají poznámky a názvy sloupců tabulky. Pak je zde DatabaseHelper, který přímo pracuje s databází a nakonec samotná databáze ***`SQLiteDatabase`***, ve které jsou data uložena. K datům v databázi se přistupuje pomocí dotazů databázového jazyka SQL. Tohoto jazyka se také využívá při vytváření samotné tabulky v databázi. Příkaz pro vytvoření tabulky je uložen v proměnné DATABASE_CREATE, ve kterém definujeme jaké sloupce má tabulka, který atribut tvoří primární klíč (id), který je v tabulce unikátní a lze podle toho jednotlivé poznámky od sebe odlišit a nalézt v tabulce. Také je definováno, že textové hodnoty nesmí být prázdné (notnull), také to že id se má vytvářet automaticky přičtením hodnoty k předchozímu přidanému id.

Pro práci s databází je nutné ji nejdříve otevřít. To se provede metodou **`public NoteBookDbAdapter open() throwsSQLException`** kde **`throws SQLException`**, znamená, že pokud nastane nějaká chyba při práci s SQL databází,

pak se v této metodě odchytí a ve výsledku se chyba neprojeví, například pádem aplikace. V této metodě se nejdříve vytvoří nový DatabaseHelper a následně se z něj získá nová databáze, do které je možné zapisovat. Při konci práci s databází, je vhodné tuto databázi uzavřít metodou `public void close()`, která uzavře samotný dbHelper (20).

Metoda `public long createNewNote(String title, String body, String filter);` Vytváří novou poznámku z předaných textových dat, jako jsou titulek, tělo poznámky a filtrovací řetězec. Tyto hodnoty se uloží do proměnné values ze třídy ContentValues. Hodnoty se pak pomocí proměnné předají hromadně metodě, která se volá na databázi a to `database.insert(DATABASE_TABLE, null, values);` Tím se vloží do databáze nová poznámka s předanými hodnotami. Tato Metoda pak vrátí číslo id vložené poznámky, takže víme pod jakým číslem se tato poznámka nachází.

`Public boolean deleteNote(long id)` maže poznámku podle id z databáze.

`Public Cursor selectAllNotes()` vrací naplněný kursor všemi poznámkami, které se nacházejí v databázi.

`Public Cursor selectNotesByFilter(String filter)` vrací kursor poznámek, ale jen těch které mají určitý filtrovací text. Toho se využívá u Spinneru v hlavní aktivitě, při výběru filtrovacích skupin.

`Public Cursor selectNote(long id) throws SQLException` vybere pouze jednu poznámku z databáze určenou číslem id.

`Public boolean updateNote(long id, String title, String body, String filter)` upravuje vybranou poznámku na nové hodnoty, které jsou předány touto metodou.

`Public Cursor selectFilters()throws SQLException` vrací kursor naplněný všemi možnými filtry, které se nacházejí v databázi. Využívá se toho pro naplnění spinneru pro filtrování.

Třída `public class DatabaseHelper` je instance této třídy je vytvořená v samotném databázovém adaptéru výše. Slouží pro práci s databází. Aby bylo možné pracovat s databází, musí dědit z `SQLiteOpenHelper`.

V konstruktoru `public DatabaseHelper(Context context)` se zavolá konstruktor nadtřídy sqlliteopenhelper (super). Zde se do konstruktoru nadtřídy předá kontext, ve kterém se má pracovat, zpravidla to je samotná aktivita, jméno databáze, a verze databáze.

Po vytvoření tohoto helperu se zavolá přepsaná metoda `public void onCreate(SQLiteDatabase db)` ve které se na předané databázi zavolá metoda, díky které se provede příkaz zadaný v textovém řetězci, který mu předáváme. V našem případě je to textový řetězec s příkazem pro vytvoření nové tabulky, definován výše. Tím se tedy v databázi vytvoří nová prázdná tabulka. Pokud dojde k upgradu tabulky, vyvolá se metoda

```
Public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion)
```

ve které se databáze vylepší na novou verzi ze staré verze. Přitom se ale musí bohužel smazat tabulka s poznámkami, které se tím ztratí a vytvoří se nová tabulka, za voláním opět metody onCreate.

4.3 Uživatelská část a vyhodnocení navržené aplikace

Pod odstavcem vidíme výsledky analýzy mé aplikace. Dále můžeme vidět poznámkový blok v emulátoru. Tedy ve virtuálním zařízení, které simuluje operační systém Google Android. Nyní bych rád vysvětlil, jak se pracuje s aplikací. Při zpuštění aplikace se objeví úvodní obrazovka, kde můžeme najít všechny poznámky, které jsme vytvořili s aktuálním datem a časem. Dále zde najdeme ikonu "+" kterou založíme novou poznámku. Další ikonou je filtrování poznámek, kde máme seřazené jednotlivé události. Poslední ikonou v úvodní obrazovce je ukončení aplikace. Když vytvoříme novou poznámku objeví se následující údaje: titulek poznámky, text poznámky. Filtr můžeme založit podle libovolně zvoleného slova nebo také připojit filtr k existující poznámce. A na konec dvě tlačítka pro potvrzení a ukončení úkonu. Když označíme jakoukoliv poznámku na úvodní obrazovce, kterou jsme si vytvořili tak aplikace přejde do prohlížení, kde můžeme vidět přesnou specifikaci dané poznámky. Najdeme zde tři ikony: zpět do úvodní obrazovky, vymazaní poznámky a editaci. Tedy má aplikace je jednoduchá, flexibilní a nenáročná na hardware telefonu. „Quick Book" nabídne například podporu starších verzí Softwaru Google Android.

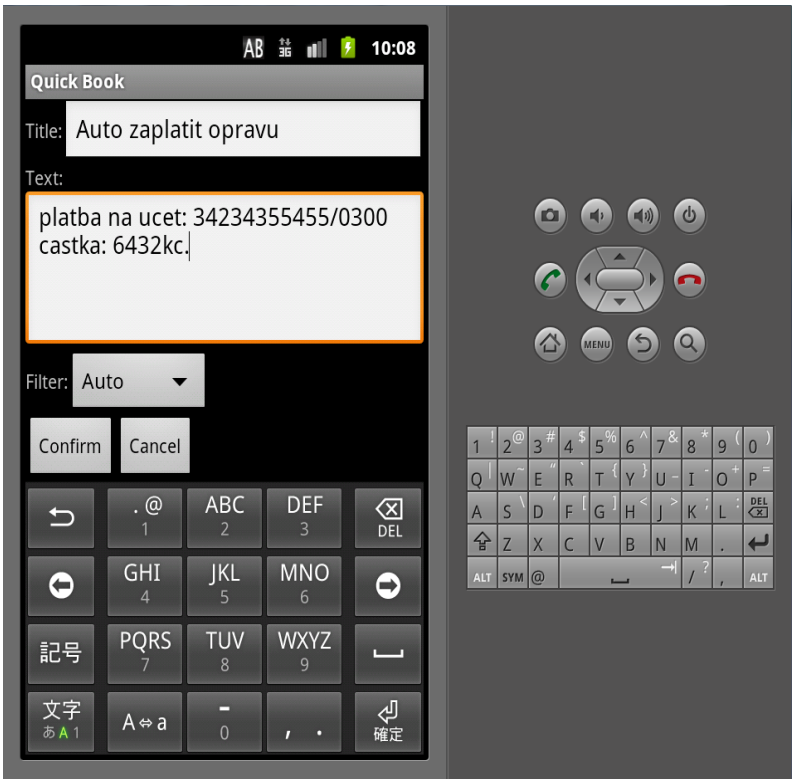
Má aplikace: „Quick Book"

Velikost aplikace	56 kb
Hardwarové vytížení systému	*CPU 17% 245 MHz, 2 MB RAM
Jednoduchost aplikace	Velmi jednoduchá
Rychlost a stabilita aplikace	Okamžitá reakce, stabilní
Filtrování poznámek	ano

Tabulka 2: Výsledné hodnoty Quick booku.

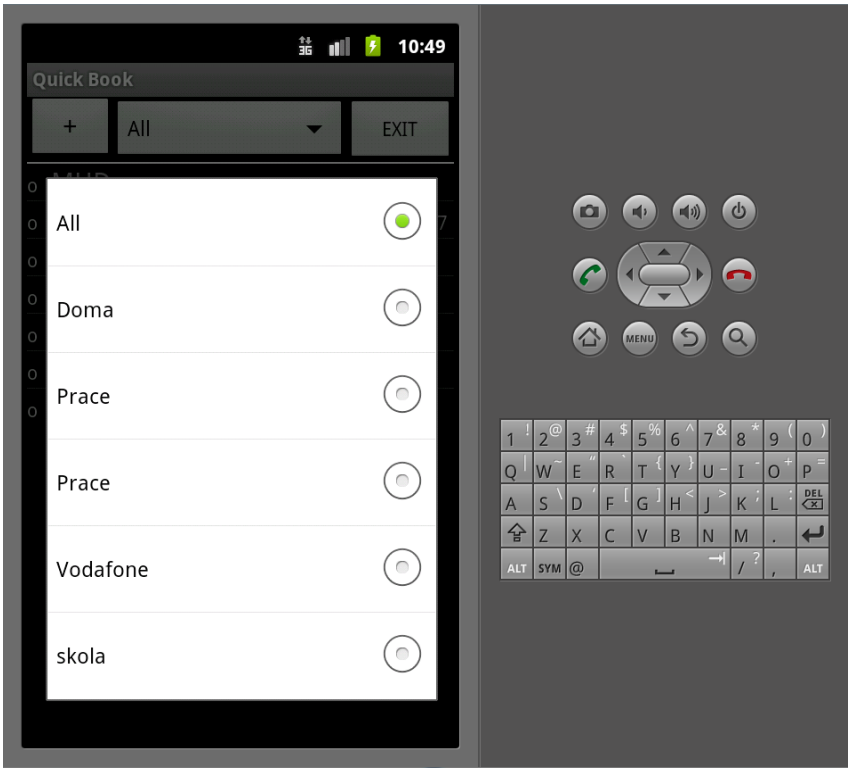
*CPU – procesor, MB RAM – operační paměť, MHz – frekvence procesoru,

Obrázek 11: Screenshot úvodní obrazovka v Quick booku.



Obrázek 12: Screenshot vytvoření poznámky v Quick Booku.

Obrázek 13: Screenshot prohlížení poznámky v Quick Booku.



Obrázek 14: Screenshot filtru poznámek.

Závěr

Google Android je zajímavým operačním systémem na poli mobilních telefonů. Díky otevřenému zdrojovému kódu přináší Android vysokou flexibilitu použití jak pro výrobce telefonů, tak i pro běžné uživatele, kteří komerčně vyvíjejí aplikace.

V teoretické části jsem popsal Globálně platformu Google Aneroid jeho architekturu a vývojové prostředí. Dále jsem porovnal vybrané nejstahovanější manažerské aplikace s Android Marketu. Cílem bylo analyzovat aplikace a podrobit je zkušebním testům a zjistit tak nedostatky těchto aplikací. A to vedlo k návrhu nové aplikace, která bude jednoduchá a flexibilní.

V praktické části se zaměřuji především na samotný vývoj aplikace, kde popisuji jak jsem danou problematiku řešil pomocí zdrojových kódů. V této části se mi podařilo naplnit stanovený cíl a to naprogramovat poznámkový blok, který je jednoduchý a má minimální zatížení na hardware telefonu. Samozřejmostí je podpora starších verzí Google Android, takže aplikaci mohou provozovat i majitelé starších telefonů. Dále se v práci zabývám vývojovým prostředím Eclipse, kde popisuji jednotlivé složky v projektu a k čemu slouží. Nakonec podrobím aplikaci analýze a popíši, jak se s ní pracuje. „Quick Book“ je určen především pro manažery ale můžou ji používat i uživatelé, kteří vlastní tento operační systém. Díky open source platformě si mohou vývojáři stáhnout zdrojový kód mé aplikace, která může být nápomocná například začínajícím programátorům apod.

V horizontu několika měsíců bych rád rozšířil aplikaci o vizuální prvky a implementoval „Quick Book“ na pozadí obrazovky neboli jako widget. Popřípadě vylepšil aplikaci o nějakou funkci navíc. Spíše budu reagovat na odezvi uživatelů s Android Marketu, kteří ohodnotí aplikaci a poskytnou mi tak zpětnou vazbu co by bylo dobré vylepšit.

Do budoucna bych se chtěl nadále zabývat problematikou programování pro mobilní platformu Google Android. Tento operační systém je velmi flexibilní a má své určité kouzlo. Osobně si myslím, že tato platforma v následujících letech dosáhne překvapivých výsledků u uživatelů.

Anotace:

Příjmení a jméno autora: Petr Svozilík

Instituce: Moravská vysoká škola Olomouc

Název práce v českém jazyce:

Softwarová ekonomicko – manažerská výbava mobilních operačních systémů

Název práce v anglickém jazyce:

Economic and Managerial Software Equipment of Mobile Operating Systems

Vedoucí práce: PhDr. Jan Lavrinčík, DiS.

Počet stran: 48

Počet příloh: 1

Rok Obhajoby: 2012

Klíčová slova českém jazyce: Android Market, Google Android, Eclipse, Java, Android software development kit (SDK)

Klíčová slova v anglickém jazyce: Android Market, Google Android, Eclipse, Java, Android software development kit (SDK)

Důvod proč jsem si vybral dané téma je, že mě tato problematika vždy zajímala. Současný stav Android Marketu jsem analyzoval a vybrané aplikace podrobil testům. Výsledkem bylo zjištění, že aplikace jsou složité a nepřehledné. Proto jsem se rozhodl učinit návrh nové aplikace, která bude jednoduchá a nijak nebude zatěžovat systém. Cílem práce bylo naprogramovat aplikaci „Quick Book“ pro manažery.

The reason why I have chosen this topic is that I was always interested about it. I have analyzed the current Android market and I have tested some applications. The applications are complicated and confusing that was the result. So I decided to make a project for a new application that will be simple and not difficult for the system. The goal was to program the application for managers called "Quick Book".

Seznam literatury a zdrojů:

(1) *Svetandroida.cz* [online]. 2012 [cit. 2012-1-13]. Každý druhý prodaný smartphone má v sobě Android. Dostupné z www: <http://www.svetandroida.cz/kazdy-druhy-prodany-smartphone-ma-v-sobe-android-201111>.

(2) *Svetandroida.cz* [online]. 2012 [cit. 2012-1-13]. Vyvíjíme pro Android. Dostupné z www: <http://www.svetandroida.cz/vyvijime-pro-android-1-uvod-201103>.

(3) *Diit.cz* [online]. 2012 [cit. 2012-1-13]. Mobilní operační systém Android. Dostupné z www: <http://diit.cz/clanek/mobilni-operacni-system-android >.

(4) *Wiki.aktualne.centrum.cz* [online]. 2012 [cit. 2012-1-13]. Open-source. Dostupné z www: <http://wiki.aktualne.centrum.cz/datarama/open-source-software>.

(5) MARK L. MURPHY,. *Android 2. Průvodce programováním mobilních aplikací.* 1. Vyd. Computer Press. 2011. ISBN: 978-80-251-3194-7.

(6) *Androidmarket.cz* [online]. 2012 [cit. 2012-1-18]. Jak vypadá Android uvnitř aneb co je ROM, kernel, bootloader a další?. Dostupné z www: <http://www.androidmarket.cz/android/jak-vypada-android-uvnitr-aneb-co-je-rom-kernel-bootloader-a-dalsi>.

(7) *Developer.android.com* [online]. 2012 [cit. 2012-2-5]. What is Android?. Dostupné z www: <http://developer.android.com/guide/basics/what-is-android.html>.

(8) *Elitecsoftware.cz* [online]. 2012 [cit. 2012-1-18]. Historie verzí platformy Android. Dostupné z www: <http://www.elitecsoftware.cz/historie-verzi-platformy-android >.

(9) *Cs.wikipedia.org* [online]. 2012 [cit. 2012-1-28]. Eclipse (vývojové prostředí). Dostupné z www: <http://cs.wikipedia.org/wiki/Eclipse_(vývojové_prostředí)>.

(10) HASHIMI, S. KOMATINENI, S. MACLEAN, D. *Pro Android 2*. 1. Vyd. Apress. 2010. ISBN-10: 1430226595.

(11) BURNETTE, E. *Hello, Android*. *Introducing Google´s Mobile Development Platform.* 1. Vyd. Progmatic Bookshelf. 2009. ISBN-10: 1934356492.

(12) MURPHY, M. *Beginning Android 2*. 1. Vyd. Apress. 2010. ISBN- 10: 9781430226291.

(13) ROGERS, R. LOMBARDO, J. MEDNIEKS, Z. MEIKE, B. *Android Application Development. Programming with the Google SDK*. 1. Vyd. O'Reilly Media. 2009. ISBN-10: 9780596521479.

(14) LAUREN DARCEY,. SHANE CONDER SAMS,. *Teach Yourself Android Application Development in 24 Hours*, 1. vyd. Apress. 2010. ISBN-10: 9780321673350.

(15) ZAKHOUR, S. HOMMEL, S. ROYAL, J. RABINOVITCH, I. RISSER, T. HOEBER, M. *Java 6 Výukový kurz*. 1. vyd. Computer Press. 2007. ISBN: 978-80-251-1575-6.

(16) *Developer.android.com* [online]. 2012 [cit. 2012-1-28]. Package Index. Dostupné z www: <http://developer.android.com/reference/packages.html>.

(17) MEIER, R. *Professional Android Application Development,* 1. Vyd. Wiley Publishing, Inc. 2009. ISBN: 978-0-470-34471-2.

(18) Wei-Meng, L. *Beginning Android Application Development,* 1. Vyd. Wiley Publishing, Inc. 2011. ISBN: 978-1-118-01711-1.

(19) *Developer.android.com* [online]. 2012 [cit. 2012-2-10]. Sample Code - Note Pad. Dostupné z www: <http://developer.android.com/resources/samples/NotePad/index.html>.

(20) *Vogella.de* [online]. 2012 [cit. 2012-2-10]. Android SQLite Database and ContentProvider - Tutorial. Dostupné z www: <http://www.vogella.de/articles/AndroidSQLite/article.html>.

Slovník zkratek:

CPU - Procesor.

MB RAM - Operační paměť.

MHz - Frekvence procesoru.

JDK - Java Development Kit.

C/C++ - Programovací jazyk.

API - Application Programming Interface, (aplikační programové rozhraní).

SDK - Software Development Kit, sada nástrojů sloužících k vývoji software.

UML - Unified Modeling Language, je v softwarovém inženýrství grafický jazyk pro vizualizaci.

HTML - Hyper Text Markup Language, je jedním z jazyků pro vytváření stránek v systému World Wide Web.

XML - Extensible Markup Language, jazyk pro strukturovanou reprezentaci dat PHP.

RSS - Je rodina XML formátů určených pro čtení novinek na webových stránkách a obecněji syndikaci obsahu.

Id - Identifikace.

PDA - Personal Digital Assistant.

OS - Operating Systém.

ROM - Read Only Memory.

SQLite - Relační databázový systém.

SSL - Secure Sockets Layer.

VPN - Virtual Private Network.

SWT - Standard Widget Toolkit.

JVM - Java Virtual Machine.

IBM - International Business Machines.

Seznam obrázků:

Seznam tabulek:

Příloha – CD

Součástí práce je cd, které zahrnuje vývojové prostředí Eclipse, Android (SDK) a (ADT), Java (JDK), aplikace Quick Book a video návod k použití.

1. Vývojové prostředí Eclipse.

2. Android software development kit (SDK).

3. Android Development Tools (ADT).

4. Java Development Kit (JDK).

5. Kompletní návod jak použit komponenty.

6. Aplikace Quick Book.